INFORMATION SYSTEM DEVELOPMENT
VOLUME 7 NO.2 (2022) | p79-83
e-ISSN 2528-5114 | p-ISSN 2477-863X
http://dx.doi.org/10.19166/xxxx

END-TO-END NEURAL NETWORK BASED CAPTCHA RECOGNITION

Jusin[1], Wilbert Harriman[2], Robin[3]

[1,3]Information System, Fakultas Ilmu Komputer, Medan, Universitas Pelita Harapan
[2]Department of Computer Science, Hsinchu City, National Tsing Hua University
E-mail: [1]jsnmpc8@gmail.com, [2]wilbert@datalab.cs.nthu.edu.tw, [3]robin80huang@gmail.com

Abstract – Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is widely used as a security measure against spam and bot attacks via the Internet. CAPTCHA works by the assumption that it takes human sensory and cognitive skills (that are not present in computers) to successfully identify objects or letters within a noisy graphical environment. In this work, we show how to build an end-to-end neural network with encoder-decoder architecture that can recognize the text in an image CAPTCHA. Our deep learning model uses a Convolutional Neural Network (CNN) encoder to convert CAPTCHA images into vector representations, followed by a Recurrent Neural Network (RNN) decoder to convert vector representations into text. Our model is able to achieve a validation accuracy of 90% after about an hour of training. Code is available at https://github.com/wilbertharriman/tf2-attention-captcha-recognizer.

***Keywords**: CAPTCHA, deep learning, neural network, supervised learning*

Abstrak – Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) telah digunakan secara luas sebagai sebuah acuan dalam melawan serangan spam dan bot melalui Internet. CAPTCHA bekerja dengan berasumsi bahwa sensori dan kognitif manusia dibutuhkan (dimana hal ini tidak dimiliki oleh komputer) agar bisa mengenal objek atau tulisan yang terdapat dalam sebuah lingkungan yang memiliki derau (noise) dengan baik dan benar. Kajian ini mengajukan sebuah cara untuk membangun end-to-end neural network dengan arsitektur encoder-decoder untuk mengenal teks pada citra CAPTCHA. Model deep learning ini menggunakan Convolutional Neural Network (CNN) encoder untuk mengkonversi citra CAPTCHA menjadi representasi vektor, kemudiaan dilanjutkan dengan menggunakan Recurrent Neural Network (RNN) decoder untuk mengkonversi representasi vector menjadi teks. Model ini mampu mencapai ketelitian validasi hingga 90% setelah dilakukan training selama 1 jam. Kode program tersedia pada alamat URL https://github.com/wilbertharriman/tf2-attention-captcha-recognizer.

Kata Kunci: *CAPTCHA, deep learning, neural network, supervised learning*

INTRODUCTION

CAPTCHA is a widely used verification method to differentiate humans from machines. CAPTCHAs can be divided into three groups based on how they interact with users: text-based, image-based, and sound-based. Our work focuses on image-based CAPTCHAs, the most common type of CAPTCHA. A typical image-based CAPTCHA involves the user seeing a noisy image, then identifying the letters in the image in the correct order. Before the advent of deep learning, computers were unable to process and learn from images in the same way a human could, which is why image-based CAPTCHAs have been so successful.

In recent years, convolutional neural networks (CNNs) have become the standard method for learning to recognize patterns from images, while recurrent neural networks (RNNs) have been shown to work well with sequential data such as text or speech. These advances motivated the combined use of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to generate high quality captions from images. Recent work proposed an attention-based model that learns to describe the content of images by learning to focus on salient objects while generating the corresponding words in the output sequence [2]. These advancements motivated the recent use of CNN-RNN architecture for generating captions for images.

Motivated by recent works, we aim to apply current advances in image captioning to build a model that is able to replicate the human ability to solve image-based CAPTCHAs. We will use "text CAPTCHA" to refer to image-based CAPTCHAs that display text. In this work, we describe how to combine convolutional neural network (CNN) and attention-based recurrent neural network (RNN) to recognize text CAPTCHAs.

The main contribution of our work is to show how image captioning can be applied to solve text CAPTCHAs, and to urge people to think twice before relying on text CAPTCHAs for their applications.

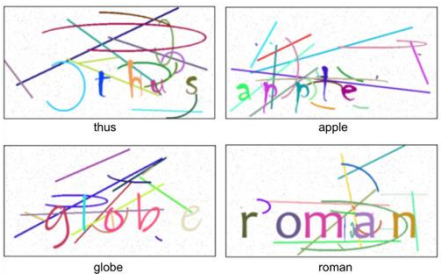


Figure 1. CAPTCHAs and their corresponding labels

DATASET

Training a deep learning model requires a large dataset. For our model, we need our training dataset to be in the form of <image, label> tuples (see Fig. 1). Our dataset consists of 100,000 labeled text CAPTCHAs for training, 20,000 for validation, as well as 20,000 unlabeled text CAPTCHAs for testing. Each image contains a word with lowercase letters.

All CAPTCHA images from the training and validation set are downsized to 50×100 RGB images. We found that downsizing CAPTCHA images to 50×100 reduces computational cost without sacrificing the quality of the CAPTCHA images. For our labels, we need to embed each letter in the alphabet as a unique token, in addition to a start token, an end token, and a token for the <space> character. Thus, we have a total of 29 unique tokens. We rely on

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

Tensorflow Keras tokenizer to handle the tokenization.

Link to the dataset can be found on our GitHub page.

ENCODER

A CNN encoder is used to transform images into meaningful representation for the decoder. Our encoder takes 50×100×3 (height × width × channel) images as inputs and turn each image into a 3D tensor of size 3×6×256, followed by a fully connected layer that transforms each image into matrices that embeds the information about the image. Our CNN consists of 6 Conv2D layers with batch normalization followed by a Dense layer.

To build a CNN that learns the desired function, we need to build a priori knowledge about the application into the network. As CAPTCHA images are noisy, we use smaller kernel sizes for our convolutional layers to build a CNN that can see ignore the noise.

DECODER

The purpose of our decoder is to output the word in a CAPTCHA image. Our decoder uses gated recurrent units (GRU) to avoid the vanishing gradient and catastrophic forgetting problems of vanilla recurrent neural networks (RNN).

When outputting letters, the decoder uses an attention layer to determine the importance of every part of an image.

END-TO-END TRAINING

In our end-to-end training step, we set our batch size to 64 and compute the cross-entropy loss between our predictions (output of our decoder) and the true labels of each batch. Our error value is then used to update the weight of both the encoder and decoder (see Fig. 2). Additionally, we train our decoder using teacher forcing [3], where ground truth from a prior time step is taken as input for faster convergence of the RNN. Lastly, we train our model for 50 epochs.

```
@tf.function
def train_step(images, targets):
    loss = 0

    hidden = decoder.reset_state(batch_size=targets.shape[0])

    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * targets.shape[0], 1)

    with tf.GradientTape() as tape:
        # features shape == (BATCH_SIZE, 18, embedding_dim)
        features = encoder(images)
        for i in range(1, targets.shape[1]):
            predictions, hidden, _ = decoder(dec_input, features, hidden)

            loss += loss_function(targets[:, i], predictions)

            # teacher forcing
            dec_input = tf.expand_dims(targets[:, i], 1)

    total_loss = loss / int(targets.shape[1])

    trainable_variables = encoder.trainable_variables + decoder.trainable_variables

    gradients = tape.gradient(loss, trainable_variables)

    optimizer.apply_gradients(zip(gradients, trainable_variables))

    return total_loss
```

Figure 2. End-to-end training step

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

Figure 3. Samples from unlabeled text CAPTCHA.

Validation

```
dataset_val = tf.data.Dataset.from_tensor_slices((img_name_val, cap_val))
dataset_val = dataset_val.map(load_image, num_parallel_calls=tf.data.experimental.AUTOTUNE)
```

```
predictions = []
targets = []

for (batch, (img, target)) in tqdm(enumerate(dataset_val)):
    img = tf.expand_dims(img, 0)
    result = evaluate(img)

    predictions.append(''.join(result))

    real_caption = ''.join([tokenizer.index_word[i] for i in cap_val[batch] if i not in [0, 1, 2]])
    targets.append(real_caption)
```

```
20000it [07:42, 43.24it/s]
```

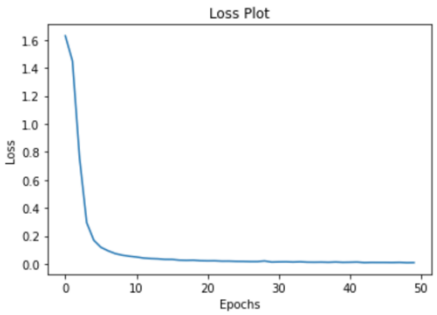
```
print('Validation accuracy:', np.mean(np.array(predictions) == np.array(targets)))
```

```
Validation accuracy: 0.90675
```

Figure 4. Validation Step

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

RESULTS AND DISCUSSION



After training for 50 epochs, we plot our loss for every epoch (see Fig. 5). We see a steep decline in our loss in the first few epochs and stable decline afterwards. This demonstrates that our end-to-end neural network is able to converge to the desired function. In the process of building our network, we found that our network has better convergence property if we initialize our weights with He uniform initializer instead of the default Xavier uniform initializer in TensorFlow.

In the validation step (see Fig. 4), we use images from our validation set and compare the output of our model to the label of the validation set. Our prediction is correct only if the whole word matches the label. The result from the validation step shows an accuracy of 90.675%.

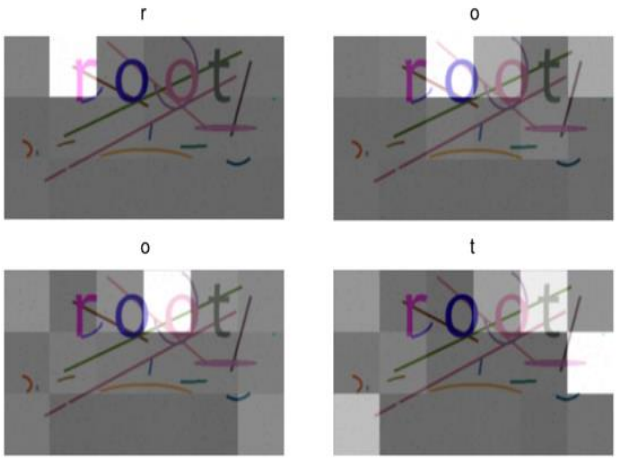


Figure 2. As the model outputs letter, its focus changes to reflect the relevant parts of the text CAPTCHA.

When we test our model on unlabeled data and sample 10 results, we obtain the following results: **[toe, that, kerry, type, mba, line, von, means, pink, kids]**.

We then verify the sampled results by ourselves (see Fig. 3). We find that all 10 predictions align with human perception.

One of the benefits of having an attention mechanism is easy visualization, we visualize the attention component learned by our model and see that our model solves CAPTCHA exactly how a human would. Our attention plot shows that the machine is able to focus on the relevant parts of the image when making prediction for each letter (see Fig. 6). Although our work focuses on lowercase letters, we can easily extend this to include support for uppercase letters, numbers or even Chinese characters as long as our dataset includes them.

CONCLUSION

In this work, we train a model that is able to correctly identify text in a text CAPTCHA with 90% accuracy. Our result shows that text CAPTCHAs are no longer secure and therefore should not be used by new applications.

REFERENCES

[1] I. GoodFellow, et al, Deep Learning, MIT Press, 2016.
[2] Xu, K., Ba, J., Kiros, R., et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Proceedings of the 32nd ICML., 2015
[3] Williams, R. J. and Zipser, D., A learning algorithm for continually running fully recurrent neural networks., *Neural computation*, 1989.

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>